New conjugacy condition with pair-conjugate gradient methods for unconstrained optimization

Abbas Y. Al-Bayati      Huda I. Ahmed
College of Computer Sciences and Mathematics
University of Mosul

Received on: 28/8/2005             Accepted on: 26/12/2005

ABSTRACT
Conjugate gradient methods are wildly used for unconstrained optimization especially when the dimension is large. In this paper we propose a new kind of nonlinear conjugate gradient methods which on the study of Dai and Liao (2001), the new idea is how to use the pair conjugate gradient method with this study (new conjugacy condition) which consider an inexact line search scheme but reduce to the old one if the line search is exact. Convergence analysis for this new method is provided. Our numerical results show that this new methods is very efficient for the given ten test function compared with other methods.

1. Introduction
We are concerned with the following unconstrained minimization problem:

minimize \( f(x) \) …(1)

where \( f : R^n \rightarrow R \) is smooth and its gradient \( g(x) = \nabla f(x) \) is exist. There are several kinds of numerical methods for solving (1), which include the steepest descent method, the Newton method and quasi-Newton methods, for example. Among them the conjugate gradient method is one choice for solving large-scale problems, because it does not need any matrices. Conjugate gradient methods are iterative methods of the form
\[ x_k = x_{k-1} + \alpha_k d_{k-1} \] \hspace{1cm} \ldots(2)

\[ d_k = \begin{cases} -g_k & \text{for } k = 1 \\ -g_k + \beta_k d_{k-1} & \text{for } k \geq 2 \end{cases} \] \hspace{1cm} \ldots(3)

where \( g_k \) denotes \( \nabla f(x_k) \) and \( \beta_k \) is a scalar.

If \( f(x) \) is a strictly convex quadratic function:

\[ f(x) = \frac{1}{2} x^T G x + b^T x + c \] \hspace{1cm} \ldots(4)

where \( G \in \mathbb{R}^{n \times n} \) is an asymmetric positive definite matrix, and \( \alpha_k \) is given by:

\[ \alpha_k = \frac{\| g_k \|^2}{d_k^T A d_k} \] \hspace{1cm} \ldots(5)

then the method (2)-(3) is called the linear conjugate gradient method, where \( \| \) denotes the Euclidean norm. The linear conjugate gradient method was originally proposed by Hestenes and Stiefel (1952) for solving linear system of equations

\[ Gx = b \] \hspace{1cm} \ldots(6)

within the framework of linear conjugate gradient methods, the conjugacy condition is defined by

\[ d_i^T G d_j = 0, \quad \text{for } i \neq j \] \hspace{1cm} \ldots(7)

for search directions, and this condition guarantees the finite termination of the linear conjugate gradient methods.

On the other hand, the method (2)-(3) is called nonlinear conjugate gradient method for general unconstrained optimization problem (general nonlinear function). The nonlinear conjugate gradient method was first proposed by Fletcher and Reeves (Fletcher and Reeves, 1964). Within the framework of nonlinear conjugate gradient methods, the conjugacy condition is replaced by

\[ d_k^T y_{k-1} = 0 \] \hspace{1cm} \ldots(8)

Where

\[ y_{k-1} = g_k - g_{k-1} \] \hspace{1cm} \ldots(9)

for search direction, because the relations:

\[ d_k^T G d_{k-1} = \frac{1}{\alpha_k} d_k^T G (x_k - x_{k-1}) = \frac{1}{\alpha_k} d_k^T (g_k - g_{k-1}) = \frac{1}{\alpha_k} d_k^T y_{k-1}. \]

Hold for the strictly convex quadratic objective function. Multiplying \( y_{k-1} \) in (3) and using (8), we can deduce a formula for the scalar \( \beta_k \), as:

\[ \beta_k = \frac{g_k^T y_{k-1}}{d_k^T y_{k-1}} \] \hspace{1cm} \ldots(10)
This is the so-called HS formula, which was given by Hestenes and Stiefel (1952), also there is well-known formulae for $\beta_k$ are the Fletcher-Reeves (FR), (Fletcher, 1964) and Polak Ribiere (PR), (Polak,1969) and (Polyak, 1969) they are given by

$$\beta_{k}^{FR} = \frac{\|g_k\|^2}{\|g_{k-1}\|^2} \quad \text{... (11)}$$

$$\beta_{k}^{PR} = \frac{g_k^T y_{k-1}}{\|y_{k-1}\|^2} \quad \text{... (12)}$$

To establish the convergence results methods mentioned above, it is usually required that the step $\alpha_k$ should satisfy the following strong Wolfe conditions

$$f(x_k + \alpha_k d_k) - f(x_k) \leq \delta \alpha_k g_k^T d_k \quad \text{... (13)}$$

$$g(x_k + \alpha_k d_k)^T d_k \leq -\sigma g_k^T d_k \quad \text{... (14)}$$

where $0 < \delta < \sigma < 1$. On the other hand, many numerical methods (e.g. the steepest descent method and quasi-Newton methods) for unconstrained optimization are proved to be convergent under the Wolfe conditions:

$$f(x_k + \alpha_k d_k) - f(x_k) \leq \delta \alpha_k g_k^T d_k \quad \text{... (15)}$$

$$g(x_k + \alpha_k d_k)^T d_k \geq \sigma g_k^T d_k \quad \text{... (16)}$$

Thus it is an important issue to study global convergence of conjugate gradient methods under the Wolfe conditions instead of the strong Wolfe conditions.

2. The Dai and Liao method

As stated in section 1, the conjugacy condition which may be represented by the form:

$$d_k^T y_{k-1} = 0 \quad \text{... (17)}$$

for nonlinear conjugate gradient methods. The extension of the conjugacy condition was studied by Peery and also Shanno (Peery,1978) and (Shanno, 1978). However, both the conjugacy conditions (7) and (17) depend on the exact line searches. In practical computation, one normally carries out inexact line search instead of exact line searches. In the case when $g_k^T d_k \neq 0$, the conjugacy conditions (7) and (17) may have some disadvantages, for this reason the extension of the conjugacy condition studied by Perry (1978), he tried to accelerate the conjugate gradient method by incorporating the secord-order information into it, specifically, he used the quasi-Newton condition

$$H_k y_{k-1} = s_{k-1} \quad \text{... (18)}$$
where $H_k$ is $n \times n$ symmetric and positive definite matrix and $s_{k-1} = \alpha_{k-1}d_{k-1}$. For quasi-Newton methods, the search direction $d_k$ can be calculated as:

$$d_k = -H_k g_k$$  \hspace{1cm} (19)

by (18) and (19), we have that

$$d_k^T y_{k-1} = -(H_k g_k)^T y_{k-1} = -g_k^T (H_k y_{k-1}) = -g_k^T s_{k-1}$$  \hspace{1cm} (20)

eq(20) is called Perry condition, which implies (17) holds if the line search is exact since, in this case $g_k^T s_{k-1} = 0$. However, practical algorithms normally adopt inexact line searches instead of exact line searches. For this reason Dai and Liao (2001) replaced the conjugacy condition (17) with the condition:

$$d_k^T y_{k-1} = -g_k^T s_{k-1}$$  \hspace{1cm} (21)

where $t \geq 0$ is a scalar. In the case $t = 0$, (21) reduces to the usual conjugacy condition (17). On the other hand, in the case $t = 1$, (21) becomes Perry’s condition (20). To ensure the search direction $d_k$ satisfies condition (21), by substituting (3) in to (21), they had obtained

$$-g_k^T y_{k-1} + \beta_k d_{k-1}^T y_{k-1} = -tg_k^T s_{k-1}$$  \hspace{1cm} (22)

this gives the Dai and Liao formula

$$\beta_k^{DL} = g_k^T (y_{k-1} - ts_k-1) / d_k^T y_{k-1}$$  \hspace{1cm} (23)

we note that the case $t = 1$ reduces to Perry formula:

$$\beta_k^P = g_k^T (y_{k-1} - s_{k-1}) / d_k^T y_{k-1}$$  \hspace{1cm} (24)

the equation (23) can be written by:

$$\beta_k^{DL} = \beta_k^{HS} - t g_k^T s_{k-1} / d_k^T y_{k-1}$$  \hspace{1cm} (25)

for which we see that formula (23) with $t \in [0, \infty)$ really defines a class of nonlinear conjugate gradient methods. Similarly, we call the method defined by (2)-(3) with $\beta_k$ from (23), method (DL), the aim of Dai and Liao is how to find the best value of $t$ to give the best nonlinear conjugate gradient method. For any $t \geq 0$, denote $d_k$ and $\overline{d}_k$ to be the search directions given by method (23) and the HS method, respectively, namely:

$$d_k = -g_k + \beta_k^{DL} d_{k-1}$$  \hspace{1cm} (26)

$$\overline{d}_k = -g_k + \beta_k^{HS} d_{k-1}$$  \hspace{1cm} (27)

Assume that $g_k^T \overline{d}_k < 0$. Then from (26), (27), (25) and $d_{k-1}^T y_{k-1} > 0$, we also have $g_k^T d_k < 0$. Thus if the direction generated by the HS method is descent, and if the line search provides the relation $d_{k-1}^T y_{k-1} > 0$, then the direction
give by DL method (23) must also be a descent direction. Denote also \( \alpha_k^* \) and \( \overline{\alpha}_k \) to be one-dimensional minimize of \( f \) along the directions \( d_k \) and \( \overline{d}_k \) respectively. Consider the following Lemma for quadratic functions (Dai and Liao, 2001).

2.1 Lemma
Suppose that \( f \) is quadratic function given in (4); then we have that:

\[
    f(x_k + \alpha_k^* d_k) - f(x_k + \overline{\alpha}_k \overline{d}_k) = \frac{(g_k^T d_{k-1})^2 t^2}{2(d_{k-1}^T G d_{k-1})(d_k^T G d_k)} \left[ \frac{2}{t} - \overline{\alpha}_k g_k^T \overline{d}_k - \frac{(g_k^T s_{k-1})^2}{s_{k-1}^T y_{k-1}} \right]
\]

... (28)

The prove of this Lemma is defined in (Dai and Liao, 2001).

from Lemma Dai and Liao obtained the best value of \( t \) which defined by:

\[
    t = \frac{g_k^T \overline{d}_k}{\tau_k}
\]

... (29)

Where

\[
    \tau_k = \overline{\alpha}_k g_k^T \overline{d}_k + \frac{(g_k^T s_{k-1})^2}{s_{k-1}^T y_{k-1}} < 0
\]

... (30)

3. New nonlinear conjugacy gradient method using pair direction

In this section we find the new value of \( t \) by using pair direction U and V, before that we give some definitions.

3.1 Definition
Vectors \( p_1, p_2, ..., p_n \in R^n \) are called left conjugate direction vectors (LCD) of a nxn real nonsingular matrix \( G \) if

\[
    \begin{cases}
    p_i^T G p_j = 0 & \text{for } i < j \\
    p_i^T G p_j \neq 0 & \text{for } i = j
    \end{cases}
\]

... (31)

that is \( P^T G P = L = ( \quad ) \),

where \( P=[p_1, p_2, ..., p_n] \). (Yuan and Golub, 2003).

3.2 Definition
Vectors \( p_1, p_2, ..., p_n \in R^n \) are called right conjugate direction vectors (RCD) of a nxn real nonsingular matrix \( G \) if

\[
    \begin{cases}
    p_i^T G p_j = 0 & \text{for } i > j \\
    p_i^T G p_j \neq 0 & \text{for } i = j
    \end{cases}
\]

... (32)

that is \( P^T G P = U = ( \quad ) \),

where \( P=[p_1, p_2, ..., p_n] \). (Yuan and Golub, 2003).
3.3 Definition

Vectors \( p_1, p_2, \ldots, p_n \in \mathbb{R}^n \) are called conjugate gradient vectors (CG) of \( n \times n \) real nonsingular matrix \( G \) if

\[
\begin{cases}
\quad p_i^T G p_j = 0 & \text{for } i \neq j \\
\quad p_i^T G p_j \neq 0 & \text{for } i = j
\end{cases}
\]

that is \( P^T GP = D = (0 \setminus 0) \), where \( P = [p_1, p_2, \ldots, p_n] \). (Yuan and Golub, 2003).

3.4 Definition

Vectors \( p_1, p_2, \ldots, p_n \in \mathbb{R}^n \) are called semi-conjugate vectors (SCD) of \( G \) if they are LCD vectors or RCD vectors of \( G \). (Yuan and Golub, 2003).

3.5 Remark

If \( G \) is symmetric and nonsingular, then we observe that the left conjugate direction vectors of \( G \) are also right direction vectors of \( G \). In this case, we call the vectors conjugate gradient vector of \( G \). In terms of Stewart’s definition (Stewart, 1973), \( U \) and \( V \) are \( G \)-conjugate if \( GUV \) is Lower triangular. Of course Stewart’s \( G \)-conjugate direction is the Left conjugate direction when \( U = V = P \). (Yuan and Golub, 2003).

3.6 Definition

Let \( G, U \) and \( V \) be nonsingular \( n \times n \) matrices. Then \( (U, V) \) is an \( G \)-conjugate pair if \( L = V^T GU \) is Lower triangular (Wyk, 1977).

3.7 New delimitative for finding the value of \( t \) for pair conjugate gradient method.

Suppose that \( f \) is given in (4). Then we have that:

\[
\begin{align*}
\quad f(x_k + \alpha_k^* v_k) - f(x_k) = & \quad [f(x_k + \alpha_k^* v_k) - f(x_k)] - [f(x_k + \alpha_k^* v_k) - f(x_k)] \\
\quad & + f(x_k) - f(x_k) \\
\end{align*}
\]

by the definitions of \( \alpha_k^* \) and \( \alpha_k^- \) for the pair direction, it is easy to show that

\[
\alpha_k^* = -\frac{g_k^T v_k}{v_k^T G v_k} \quad \text{and} \quad \alpha_k^- = -\frac{g_k^T v_k}{v_k^T G u_k}
\]

and by the strong Wolfe condition (13), we have

\[
[f(x_k + \alpha_k d_k) - f(x_k)] \leq \delta \alpha_k g_k^T d_k
\]

then (34) becomes

\[
\begin{align*}
\quad f(x_k + \alpha_k^* v_k) - f(x_k + \alpha_k^* u_k) & \leq \delta \left( -\frac{g_k^T v_k}{v_k^T G v_k} \right) g_k^T v_k - \delta \left( -\frac{g_k^T v_k}{v_k^T G u_k} \right) g_k^T u_k + f(x_k) - f(x_k) \\
\end{align*}
\]

… (36)
since $0 < \delta < 1$ then we can take $\delta = \frac{1}{2}$,
\[
f(x_k + \alpha_k v_k) - f(x_k) = \frac{1}{2} \left( -g_k^T v_k \right) g_k^T v_k + \frac{1}{2} \left( -g_k^T v_k \right) g_k^T u_k + f(x_k) - f(x_k^*)
\]

\[
= \frac{1}{2} \left( g_k^T v_k \right) g_k^T u_k - \frac{1}{2} \left( g_k^T v_k \right) \left( g_k^T v_k \right) + f(x_k) - f(x_k^*)
\]

\[
= \frac{1}{2(v_k^T G v_k)(v_k^T G u_k)} + f(x_k) - f(x_k^*)
\]

where
\[
\Gamma = \left( g_k^T v_k \right) \left( g_k^T v_k \right) - \left( g_k^T v_k \right) \left( v_k^T G u_k \right)
\]

since
\[
v_k = -g_k \]
\[
u_k = v_k + \beta_k^{DL} u_{k-1}
\]

Where $\beta_k^{DL}$ is defined in (25) [$\beta_k^{DL} = \beta_k^{HS} - \frac{\beta_k^{HS} s_{k-1}}{d_{k-1} s_{k-1}}$]

and define $\mu_k = \frac{-s_k^T s_{k-1}}{u_{k-1}^T y_{k-1}}$

then (42) becomes
\[
u_k = v_k + \beta_k^{HS} u_{k-1} + \mu_k u_{k-1}
\]

Now substitute (45) in (40) to get
\[
\Gamma = \left( g_k^T v_k \right) \left( v_k^T G v_k \right) \left( g_k^T v_k \right) \left( v_k^T G u_k \right) - \left( g_k^T v_k \right) \left( v_k^T G u_k \right)
\]

since $(v_k^T G u_{k-1}) = 0$ (from the definition of the semi conjugate direction)

then (46) becomes:
\[
\Gamma = \left( g_k^T v_k \right) \left( v_k^T G v_k \right) \left( g_k^T v_k \right) \left( v_k^T G u_k \right) - \left( g_k^T v_k \right) \left( v_k^T G v_k \right)
\]

from (39) we have
Abbas Y. Al-Bayati and Huda I. Ahmed

\[
\begin{align*}
\beta_k^{HS} g_k^T u_{k-1} + t \mu_k g_k^T u_{k-1} &= f(x_{k}) - f(x_{k-1}) + \frac{2(v_k^T G v_k)}{g_k^T v_k} \\
(\beta_k^{HS} + t \mu_k)(g_k^T u_{k-1}) &= f(x_{k}) - f(x_{k-1}) + \frac{2}{-\alpha_k} \\
(\beta_k^{HS} + t \mu_k) &= f(x_{k}) - f(x_{k-1}) \left( \frac{2}{-\alpha_k} \right) \\
t \mu_k &= f(x_{k}) - f(x_{k-1}) \left( \frac{2}{-\alpha_k} \right) - \beta_k^{HS} \\
t &= f(x_{k}) - f(x_{k-1}) \left( \frac{2(u_k^T y_{k-1})}{\alpha_k (g_k^T g_{k-1})} \right) + \frac{(g_k^T y_{k-1})}{(g_k^T s_{k-1})} \\n\end{align*}
\]

where \( t > 0 \) is a scalar. In practical if we have to take \( \tilde{t} = \frac{1}{t} \) which give the best result, then the new formula for the pair conjugate gradient method is defined by:

\[
\beta_k^{New} = \beta_k^{HS} - \tilde{t} \left( \frac{g_k^T s_{k-1}}{u_k^T y_{k-1}} \right) 
\]

we call this new formula (49) with (2)-(3) by the new pair method.

4. The algorithm of the new pair conjugate gradient method

We list below the outlines of the new method

For an initial point \( x_0 \):

**Step (1):** set \( k = 1 \), \( v_{k-1} = -g_{k-1} \).

**Step (2):** set \( x_k = x_{k-1} + \alpha_k v_{k-1} \), where \( \alpha_k \) is a scalar chosen in such a way such that \( f_k < f_{k-1} \).

**Step (3):** check for convergence, i.e. if \( |f_k| \leq \epsilon \), where \( \epsilon \) is small positive tolerance, stop; otherwise continue.

**Step (4):** if \( k \geq 2 \) go to step (5), else go to step (8).

**Step (5):** compute \( x_k = x_{k-1} + \alpha_k u_k \), where \( \alpha_k = -\alpha_k \left( \frac{g_k^T v_k}{u_k^T (g_{k-1} - g_k)} \right) \).

**Step (6):** check for convergence, i.e. if \( |g_{k+1}| \leq \epsilon \), where \( \epsilon \) is small positive tolerance, stop; otherwise continue.
Step (7): compute the value of $\hat{t}$ where $\hat{t}$ becomes

$$\hat{t} = 1 \left[ f(x_k) - f(x_{k+1}) \right] = - \left( \frac{2(u_{k-1}^Ty_{k-1})}{\alpha_k (g_k^T u_{k-1})(g_k^T s_{k-1})} \right) + \left( g_k^T y_{k-1} \right).$$

Step (8): Compute the new search direction $u_k = -g_k + \beta_k^{new} u_{k-1}$, where $\beta_k$ is computed by the following formula

$$\beta_k^{new} = \beta_k^{HS} - \frac{g_k^T s_{k-1}}{u_{k-1}^T y_{k-1}}.$$

Step (9): if $k=n$ or if $\|s_k^T g_{k-1}\|>0.2\|g_k\|^2$ is satisfied go to step (1), else, set $k=k+1$, and go to step (2).

5. Generalized conjugate directions

We will now formulate the analogous generalized conjugate direction method for the minimization of function $f(x)$. Suppose that $U$ and $V$ form a conjugate pair. Set $x_0 = \text{arbitrary}$, $g_0 = g(x_0)$, for I=0,1,---,compute:

$$x_{k+1} = x_k + \alpha_k v_k$$

where $\alpha_k$ minimizes $f(x_k + \alpha_k v_k)$ as a function of $\alpha$, and let

$$g_k = g(x_k), \quad g_{k+1}^* = g(x_{k+1})$$

$$\alpha_k^* = -\alpha_k \left[ \frac{g_k^T v_k}{u_k^T (g_{k+1} - g_k)} \right]$$

$$x_{k+1} = x_k + \alpha_k^* u_k$$

Before we prove that this algorithm will find the minimum of quadratic function in $n$ steps, then we show that if $f$ is quadratic then $\alpha_k^*$ in (35) are the same as the $\alpha_k^* = -\frac{g_k^T v_k}{v_k^T G u_k}$, in fact

$$\alpha_k^* = -\alpha_k \left[ \frac{g_k^T v_k}{u_k^T (g_{k+1} - g_k)} \right] = \alpha_k \left[ \frac{v_k^T (g_{k+1} - g_k)}{u_k^T (g_{k+1} - g_k)} \right]$$

$$= \alpha_k \left[ \frac{v_k^T G (x_{k+1}^* - x_k)}{u_k^T G (x_{k+1}^* - x_k)} \right] = \alpha_k \left[ \frac{1}{\alpha_k} \frac{v_k^T G (x_{k+1}^* - x_k)}{u_k^T G (x_{k+1}^* - x_k)} \right]$$

$$= \alpha_k \left[ \frac{v_k^T G v_k}{u_k^T G v_k} \right] = \alpha_k \left[ \frac{v_k^T G v_k}{u_k^T v_k} \right] = \alpha_k$$

(see, Wyk, 1977).
6. Theorem

If the iteration (50) is applied to the quadratic function where 
(U,V) form a G-conjugate pair, the minimum is found in at most n 
iterations, moreover, x_n lies in the subspace generated by x_0 and 
v_{i_0}, v_{i_1}, \ldots, v_{i_{n-1}}.

Proof:
The first result is established by proving that 
g^T_i v_j = 0
for all i<n and j=0,1,\ldots,i, by induction. For i=0,
g^T_i v_0 = (Gx_0 + b)^T v_0 = [G(x_0 + \alpha^{**}_i u_0 + b)]^T v
\quad = g^T_0 v_0 + \alpha^{**}_i u^T_0 Gv_0
\quad = g^T_0 v_0 - (\frac{g^T_0 v_0}{v^T_0 G u_0})u^T_0 Gv_0 = 0

now suppose that 
g^T_i v_j = 0
for some i and j=0,1,\ldots,i-1, then for j=0,1,\ldots,i
\quad g^T_i v_j = g^T_i v_j + \alpha^{**}_i u^T_i G u_i.

Due to the induction hypothesis and the conjugacy,
g^T_i v_j = 0 = v^T_i G u_i
for all j<i, and for the case j=i
\quad g^T_i v_i - (\frac{v^T_i G u_i}{v^T_i G u_i})v^T_i G u_i = 0 , (see, Wyk, 1977).

7. Numerical results

We tested the HS method (10), Perry method (24), DL method (25) 
and our new pair conjugate gradient method (49) All results are obtained 
using Pentium 4 workstation and all programs are written in Fortran 
language. Our line search subroutine computes $\alpha_k$ such that the strong 
Wolfe condition (13)-(14) hold with $\delta = 0.001$ and $\sigma = 0.9$. The initial value 
of $\alpha_k$ is always compute by a cubic fitting procedure which was described 
details by Bunday (Bunday, 1982) used as a line search procedure. 
Although our line search cannot always ensure the descent property of $d_k$ 
for all three methods, uphill search directions seldom occur in our numerical 
experiments. In the case when an uphill search direction does occur, we 
restart the algorithm by setting $d_k = -g_k$. For the DL method (25) $\tau = 0.1$ is 
selected. (see Dai and Liao, 2001).
We have tested ten functions with different dimensions n=100, 1000 and 10000. The numerical results are given in the form of NOF and NOI where NOF denote the numbers of function evaluations, and NOI denote the numbers of iterations. The stopping condition is $\|s_{k+1}\| \leq 1 \times 10^{-5}$.

Comparing the new pair method (49) with HS method, Peery method, DL method, we could say that the new method is more efficient than all especially for Powell function, Wood function, Helical function, Powell3 function, Edger function and Recip function from the ten function test in this section as we see from the Table (7.1), (7.2), (7.3).

**Table (6.1A)**

<table>
<thead>
<tr>
<th>Function</th>
<th>NOF</th>
<th>NOI</th>
<th>NOF</th>
<th>NOI</th>
<th>NOF</th>
<th>NOI</th>
<th>NOF</th>
<th>NOI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Powell</td>
<td>180</td>
<td>60</td>
<td>131</td>
<td>48</td>
<td>143</td>
<td>49</td>
<td>123</td>
<td>40</td>
</tr>
<tr>
<td>Wood</td>
<td>103</td>
<td>49</td>
<td>103</td>
<td>49</td>
<td>103</td>
<td>49</td>
<td>71</td>
<td>25</td>
</tr>
<tr>
<td>Powell3</td>
<td>43</td>
<td>20</td>
<td>32</td>
<td>15</td>
<td>48</td>
<td>23</td>
<td>35</td>
<td>14</td>
</tr>
<tr>
<td>Helical</td>
<td>250</td>
<td>123</td>
<td>246</td>
<td>121</td>
<td>250</td>
<td>123</td>
<td>82</td>
<td>33</td>
</tr>
<tr>
<td>Edger</td>
<td>16</td>
<td>6</td>
<td>14</td>
<td>5</td>
<td>16</td>
<td>6</td>
<td>15</td>
<td>6</td>
</tr>
<tr>
<td>Recip</td>
<td>31</td>
<td>11</td>
<td>27</td>
<td>10</td>
<td>31</td>
<td>11</td>
<td>16</td>
<td>5</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>623</td>
<td>269</td>
<td>553</td>
<td>248</td>
<td>591</td>
<td>261</td>
<td>342</td>
<td>123</td>
</tr>
</tbody>
</table>

**Table (6.1B)**

Performance Percentage for the new pair CG algorithm compared with others and for n=100

<table>
<thead>
<tr>
<th>Tools %</th>
<th>HS method</th>
<th>Perry method</th>
<th>DL method</th>
<th>New method</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOF %</td>
<td>100 %</td>
<td>86</td>
<td>95</td>
<td>55</td>
</tr>
<tr>
<td>NOI %</td>
<td>100 %</td>
<td>92</td>
<td>97</td>
<td>55</td>
</tr>
</tbody>
</table>

**Table (6.2A)**

<table>
<thead>
<tr>
<th>Function</th>
<th>NOF</th>
<th>NOI</th>
<th>NOF</th>
<th>NOI</th>
<th>NOF</th>
<th>NOI</th>
<th>NOF</th>
<th>NOI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Powell</td>
<td>219</td>
<td>66</td>
<td>131</td>
<td>48</td>
<td>143</td>
<td>49</td>
<td>140</td>
<td>41</td>
</tr>
<tr>
<td>Wood</td>
<td>103</td>
<td>49</td>
<td>103</td>
<td>49</td>
<td>103</td>
<td>49</td>
<td>77</td>
<td>27</td>
</tr>
<tr>
<td>Powell3</td>
<td>49</td>
<td>23</td>
<td>35</td>
<td>16</td>
<td>52</td>
<td>25</td>
<td>35</td>
<td>14</td>
</tr>
<tr>
<td>Helical</td>
<td>270</td>
<td>133</td>
<td>268</td>
<td>123</td>
<td>272</td>
<td>134</td>
<td>82</td>
<td>33</td>
</tr>
<tr>
<td>Edger</td>
<td>18</td>
<td>7</td>
<td>17</td>
<td>6</td>
<td>18</td>
<td>7</td>
<td>15</td>
<td>6</td>
</tr>
<tr>
<td>Recip</td>
<td>33</td>
<td>12</td>
<td>33</td>
<td>12</td>
<td>33</td>
<td>12</td>
<td>16</td>
<td>5</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>692</td>
<td>290</td>
<td>587</td>
<td>254</td>
<td>621</td>
<td>276</td>
<td>365</td>
<td>126</td>
</tr>
</tbody>
</table>
Table (6.2B)
Performance Percentage for the new pair CG algorithm compared with others and for n=1000

<table>
<thead>
<tr>
<th>Tools</th>
<th>HS method</th>
<th>Perry method</th>
<th>DL method</th>
<th>New method</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOF %</td>
<td>100 %</td>
<td>92</td>
<td>90</td>
<td>53</td>
</tr>
<tr>
<td>NOI %</td>
<td>100 %</td>
<td>88</td>
<td>95</td>
<td>43</td>
</tr>
</tbody>
</table>

Table (6.3A)
Numerical comparisons of the new CG method with n=10000

<table>
<thead>
<tr>
<th>Function</th>
<th>NOF</th>
<th>NOI</th>
<th>NOF</th>
<th>NOI</th>
<th>NOF</th>
<th>NOI</th>
<th>NOF</th>
<th>NOI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Powell</td>
<td>253</td>
<td>72</td>
<td>133</td>
<td>49</td>
<td>178</td>
<td>57</td>
<td>186</td>
<td>47</td>
</tr>
<tr>
<td>Wood</td>
<td>105</td>
<td>50</td>
<td>105</td>
<td>50</td>
<td>105</td>
<td>50</td>
<td>77</td>
<td>27</td>
</tr>
<tr>
<td>Powell3</td>
<td>51</td>
<td>24</td>
<td>37</td>
<td>17</td>
<td>52</td>
<td>25</td>
<td>35</td>
<td>14</td>
</tr>
<tr>
<td>Helical</td>
<td>249</td>
<td>145</td>
<td>290</td>
<td>143</td>
<td>294</td>
<td>145</td>
<td>82</td>
<td>33</td>
</tr>
<tr>
<td>Edger</td>
<td>18</td>
<td>7</td>
<td>17</td>
<td>6</td>
<td>18</td>
<td>7</td>
<td>15</td>
<td>6</td>
</tr>
<tr>
<td>Recip</td>
<td>33</td>
<td>12</td>
<td>33</td>
<td>12</td>
<td>33</td>
<td>12</td>
<td>16</td>
<td>5</td>
</tr>
<tr>
<td>Total</td>
<td>709</td>
<td>310</td>
<td>615</td>
<td>277</td>
<td>680</td>
<td>296</td>
<td>411</td>
<td>123</td>
</tr>
</tbody>
</table>

Table (6.3B)
Performance Percentage for the new pair CG algorithm compared with others and for n=10000

<table>
<thead>
<tr>
<th>Tools</th>
<th>HS method</th>
<th>Perry method</th>
<th>DL method</th>
<th>New method</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOF %</td>
<td>100 %</td>
<td>87</td>
<td>96</td>
<td>60</td>
</tr>
<tr>
<td>NOI %</td>
<td>100 %</td>
<td>89</td>
<td>95</td>
<td>40</td>
</tr>
</tbody>
</table>
Appendix:

These test functions are famous and form general literature

1- Generalized Powell function:

\[
f(x) = \sum_{i=1}^{n/4} \left[ (x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4 \right]
\]

\[x_0 = (3,-1,0,1,...)^T .\]

2- Generalized Wood function:

\[
f(x) = \sum_{i=1}^{n/4} 100(x_{4i-2} - x_{4i-3})^2 + (1 - x_{4i-3})^2 + 90(x_{4i} - x_{4i-1})^2 + (1 - x_{4i-1})^2
\]

\[+ 10.1((x_{4i-2} - 1)^2 + (x_{4i} - 1)^2) + 19.8(x_{4i-1} - 1)(x_{4i} - 1),\]

\[x_0 = (-3,-1,-3,-1,...)^T .\]

3- Generalized Edeger function:

\[
f(x) = \sum_{i=1}^{n/4} [(x_{2i-1} - 2)^4 + (x_{2i-2} - 2)^2 * x_{2i}^2 + (x_{2i} + 1)^2],
\]

\[x_0 = (1,0,...)^T .\]

4- Generalized Powell3 function:

\[
f(x) = \sum_{i=1}^{n/4} \left\{ \left[ 3 - \frac{1}{1 + (x_i - x_{2i})^2} \right] - \sin \left( \frac{\pi x_{2i} x_{3i}}{2} \right) \right\} \exp \left[ -\left( \frac{x_i + x_{3i} - 2}{x_{2i}} \right)^2 \right],
\]

\[x_0 = (0,1,2,...)^T .\]

5- Generalized Helical function:

\[
f(x) = 100(x_{3i}) - 100 \left( \frac{1}{2\pi} a \tan \left( \frac{x_{3i-1}}{x_{3i-2}} \right) \right)^2 + 100 \left( \frac{x_{3i-2} + x_{3i-1}^2}{x_{3i-2}} \right)^2 + x_{3i}^2
\]

\[x_0 = (-1,0,0,...)^T .\]

6- Generalized Recip function:

\[
f(x) = \sum_{i=1}^{n/4} \left\{ (x_{3i-1} - 5)^2 + x_{3i}^2 + \frac{x_{3i}^2}{(x_{3i-1} - x_{3i-2})^2} \right\},
\]

\[x_0 = (2,5,1,...)^T .\]

REFERENCES


